Building Large Scale Mosaics from Landsat Data

Lucian Plesea
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive, MS 126-234
Pasadena, CA 91109-8099
818 354-3928
Lucian.Plesea@jpl.nasa.gov

Joseph Jacob
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive, MS 126-234
Pasadena, CA 91109-8099
818 354-0673
Joseph.Jacob@jpl.nasa.gov

ABSTRACT

This paper describes the process and tools used for creating a full resolution seamless Landsat mosaic of the continental U.S. at 30 meters per pixel. The 6-band, 150GB image was built from 428 individual Landsat scenes of the Multi-Resolution Land Characteristics (MRLC) dataset. Custom designed software based on SGI ImageVision framework was developed on JPL supercomputers to accomplish this task as a single step operation, with minimal operator input. Novel techniques were used to deal with data storage and access issues. Various color matching algorithms can be used to generate a parameter set, which in turn controls all the details of building the mosaic. Some of the tested algorithms and their results will also be discussed.

Keywords

Mosaic, Landsat, color matching, supercomputer.

1. INTRODUCTION

With the advent of Landsat [1] imagery, it became relatively easy to obtain high quality imagery for large portions of the globe. Individual Landsat scenes and mosaics of a few scenes are routinely used in various applications [4]. Attempting to assemble more than a few scenes, especially across a number of Landsat orbital paths is a complex task. Six Federal environmental monitoring programs, EMAP (US EPA), GAP (USGS), NAWQA (USGS), C-CAP (NOAA), NALC (US EPA/USGS), and the RSA Center (USFS) have formed a partnership with the EROS Data Center (USGS) in 1995 to facilitate the development of comprehensive land characteristics information for the United States. This partnership established the Multi-Resolution Land Characteristics Interagency Consortium (MRLC) [2] to collectively purchase a set of Landsat scenes to serve as a basis for nation-wide land coverage studies. 530 scenes collected in 1992 and 1993 were selected for minimal cloud coverage, converted to UTM [3] projection, geo-referenced and correlated with the USGS Level 1 Digital Elevation Model.

Since JPL and Caltech supercomputing assets are frequently used for large scale visualizations, we became very interested in this dataset. The potential of applying supercomputing to build a full resolution Landsat image mosaic of the continental U.S. from this dataset was recognized.

2. REQUIREMENTS

Finding the right brightness corrections to be applied to each scene so that they can be blended into a mosaic can be done using a multitude of algorithms. The ability to experiment with various approaches was considered essential. To accommodate this, a deliberate decision was made to separate the brightness correction algorithm from the mosaicking code. A set of parameters that encapsulate all the information needed to build a mosaic can be generated by an external process and then used to build a mosaic.

Since manipulating such large quantities of data takes considerable time, even on a high performance supercomputer, a very important requirement is the ability to stop and restart the generation of the mosaic at any point. Also, it is essential to be able to regenerate a small portion of the mosaic without affecting the rest of the image. On our supercomputer, most of the available computing time is available via a job queuing system, so the ability to run in non-interactive, unsupervised mode was essential.

Severe restrictions were also imposed by the very large dataset. A single step process, from raw input files to the final mosaic eliminates the storage of intermediate data and guarantees the propagation of any change in input data and parameters to the output.

3. APPROACH

Since JPL's main supercomputer is a SGI Origin 2000, the development environment is very well suited for image processing tasks. In particular, SGI's ImageVision Library (IL) [5] provides a rich framework, with features such as parallel processing, integrated memory management, flexible input/output storage options and a large pre-built set of image processing operators. Using the IL requires building an image processing chain from available and custom modules, and then asking for areas of the output image. The processing engine follows the dependency chain and assigns processing units to modules in the proper order to generate the requested output. Being a C++ object library, it is also easily extendable with custom operators. The main disadvantage is that the IL has not been ported to any environment other than SGI IRIX.

3.1 Mosaicking Process Description

The output image is generated in smaller areas (tiles). This allows for start/stop/restart at any tile boundary without affecting the rest of the mosaic. For each tile, a subset of the input dataset that could contribute to the output tile is selected. This reduces drastically the size of the input dataset that needs to be used for each tile, and also introduces an important mechanism for controlling the processing efficiency. A larger output tile size uses more input images but utilizes the cache better, while a smaller tile incurs a much larger I/O overhead. Within each tile, smaller regions (pages) are used as the unit of data, each one being computed in a potentially parallel environment.

The input images are first remapped to the output projection using a choice of bilinear or cubic interpolation. This is done via an IL custom warp operator that implements the UTM to Platte Carre (latitude and longitude) conversion. The last step is to blend all the transformed and color corrected input images into the output tile via a custom weighted sum blend operator. This operator applies the brightness correction to the input images independently for every band, then computes a weighted sum between all the input images, using the pixel values from a blend mask image (described later) as the weights. For example, let j be the input image (j from 1 to N), and i the band (i from 1 to 6). For each pixel value K_i in the band i of the output mosaic, let P_{ij} be the value of the corresponding pixel value from image j band i. Let G_{ij} and O_{ij} be the multiplicative and additive corrections for image j band i, and Q_j be the value of the blend mask for image j. Using this notation, the value of the output pixel can be computed using the following formula.

$$K_i = \frac{\sum_{j=1}^{N} \left[Q_j * \left(G_{ij} * P_{ij} + O_{ij} \right) \right]}{\sum_{j=1}^{N} Q_j}.$$

The output image built using this algorithm is then written to the output file.

The values for Q_j for each input image are obtained from a corresponding blend mask image. This mask, which follows the image through the projection transformation, is used to produce a smooth blended edge where scenes overlap and also to eliminate undesirable portions of the input image. The use of a blend mask solves a multitude of problems. A Landsat scene in UTM projection does not fill the whole rectangular input image since the satellite has an 8 degree inclination. A blend mask value of 0 in the areas known not to be covered will effectively eliminate those input pixels from the output. The overlap between Landsat scenes is also hard to determine. Due to the geometry of the trajectory, the overlap varies between a nominal 7.3% at the equator and 80% at 81 degrees latitude. Orbit to orbit variability, orthorectification and georeferencing of an image introduce even more uncertainty in the precise location of a Landsat scene edge.

3.2 Brightness Correction

As described, the brightness matching only allows for a multiplicative and an additive factor to control the brightness correction for each band within each scene. This produces reasonable output but does not allow for corrections for areas smaller than the scene. A better approach would be to have a correction mask for each band within each scene, thus gaining precise control over the value of each individual input pixel. This seems like a lot of storage overhead, but in most generic cases these masks will be rather uniform, subject to good compression ratios and parametric description. For example, our blend masks compress to less than 1% of a scene. It should be noted that if the blend masks and the correction factors are not present, default values are used, so that uncorrected mosaics can be generated for test purposes.

With the framework described above a large number of possible algorithms for brightness corrections can be tested. A statistical approach, using only the distribution of values within every band of every input scene is simple to implement and efficient to run, since the amount of data to be analyzed is much reduced. One of the early approaches was to force all the scenes to have the same average brightness. This of course produced a seamless mosaic, however it looked very unnatural since it had no large-scale variations of color. Forcing a uniform brightness over the chlorophyll band (usually mapped to visual green), and then using the multiplicative and additive factors determined in this way across all the other bands, followed by limiting the variations between neighbor scenes produced a much better result. Since most of the brightness information is obtained from the green component, this approach generated a uniformly bright image, yet permitted natural variations of color. One of the main problems common to all the statistic-based algorithms was introduced by the coastal scenes, which had a natural very low average brightness. This was solved by ignoring the low intensity pixels within the image, in effect only taking into account reasonably reflective land areas.

Another possible approach used to determine the appropriate brightness correction is based on finding the best brightness match in the overlapping area. Each pixel P_i in the overlap region corresponds to pixel value X_i in one scene and Y_i in the neighboring scene. For different pixels, X_i may be greater than, less than or equal to Y_i, but taken collectively the brightness of corresponding pixels in the overlap region of two Landsat scenes tends to have a linear relationship. One exception is the case of clouds, which may appear in one scene and not in the other.

To determine the appropriate additive and multiplicative brightness correction to match a scene with its neighbor, we first generate a scatter plot by plotting the pixel values in one scene, X_i, versus the pixel values in the neighboring scene, Y_i, for all pixels P_i in the overlap region. A least squares approach [6] is then used to determine the best line fit through the plotted points. The ideal best line fit to match the brightness for the two scenes is $y=x$. If the actual best line fit is determined to be $y=mx+b$, the appropriate additive correction is $-b$ and multiplicative correction is $1/m$.

The least squares approach ensures that all residual errors fit within a certain tolerance. Outlying points are eliminated and do not contribute to the final solution. This is an important advantage that permits matching of scenes with cloud cover, because pixels that correspond to clouds that appear in one scene and not in the other tend to become outliers in the least squares solution.

3.3 Data Storage Format

ImageVision Library supports a large selection of image file formats, but none of the existing ones provided all the features we needed. However, since it is easily extendible, a custom file format was designed and implemented. A previously existing format, providing hierarchical, multi-spectral and paged image storage was used as the starting point. This format was also greatly enhanced to supply additional features. The pre-existing format specified the order of the pages within the file, the offset at which each page is stored being uniquely determined by a linear combination of resolution level, page corner position within the whole image and spectral channel.

To make this format more flexible, one level of indirection between the page file offset and the page location within the image was introduced. This was done by the introduction of a separate index file, containing a small record for every page. This record specifies the storage size of the page and the offset of the page within the data file. The order of records in the index file is predetermined, the same as the page order in the original image file format. This addition makes possible per-page compression since the size of a page is now flexible, and also adds efficient sparse image storage. We support both JPEG and LZW image compression since both are readily available and are used extensively. In addition to standard versions of these, a few custom variations were attempted, but they are still in the early development phase.

Another useful feature of this image file format is the fact that the image file is consistent as soon as an empty data file and a zero filled index file is created. A record containing a size of zero specifies that the whole page is black, and no additional storage space is required. A newly created file, with a zero size data file and an index file filled with zeros is thus a perfectly valid black image. As pages are written or modified, they are added at the end of the file, and the record in the index file is populated. Thus, the output file can be opened for inspection while it is being created, allowing early detection of potential problems and an easy visual inspection of the generated images. Using this format, the image may be modified by simply appending the new pages of image data to the data file and modifying the corresponding records in the index file. If a copy of the previous index file is made and preserved, simple access to both versions of the file is possible at a minimal storage cost.

4. CONCLUSION

A simple but powerful framework for producing seamless large area mosaics has been presented. This code was successfully used to produce a one arc second, six band Landsat image of the whole continental U.S. This image is 215,000 pixels wide by 95,000 pixels high and takes 150GB of storage space in compressed form. Building this image took 10 hours on 32 CPUs of an Origin 2000, while using 4 GB of memory.

The flexibility of the approach makes it a good candidate for other data sources and also for even larger mosaics. The image file format built and used for this mosaic has several advantages over other file formats, and could also be used by other large visual data storage applications. The result of this process is available on the web at http://mapus.jpl.nasa.gov/.

5. ACKNOWLEDGMENTS

The work described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Air Force Research Laboratory, Warfighter Training Research Division through an agreement with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, or the Jet Propulsion Laboratory, California Institute of Technology.

6. REFERENCES

[1] Landsat 7 Documents, http://landsat7.usgs.gov/resource.html.

[2] MRLC Homepage, http://www.epa.gov/docs/grd/mrlc.

[3] Richardus, P. and Adler, R., Map Projections, ISBN 0444 103627, p. 137.

[4] Schowengerdt, R.A., Remote Sensing, Models and Methods for Image Processing, Second Edition, ISBN 0-12-628981-6.

[5] SGI ImageVision Library, http://www.sgi.com/software/imagevision.

[6] Sokolnikoff, I.S. and Redheffer, R.M., Mathematics of Physics and Modern Engineering, ISBN 07-059625.